32장 String

32.1 String 생성자 함수

- String 객체는 생성자 함수 → new 연산자와 함께 호출하여 String 인스턴스를 생성할 수 있다.

```
const strObj = new String();
console.log(strObj); // String {length:0, [[PrimitiveValue]]:""}
```

- String 생성자 함수에 인수를 전달하지 않고 new 연산자와 함께 호출하면 [[StringData]] 내부 슬롯에 빈 문자열을 할당한 String 래퍼 객체를 생성한다.

```
const strObj = new String('Lee');
console.log(strObj); // String {0:"L", 1:'e", 2:"e", length:3, [[PrimitiveValue]]:"Lee"}
```

- String 생성자 함수의 인수로 문자열을 전달하면서 new 연산자와 함께 호출하면 [[StringData]] 내부 슬롯에 인수로 전달받은 문자열을 할당한 String 래퍼 객체를 생성한다.

```
const strObj = new String('Lee');
console.log(strObj[0]); //L

strObj[0] ='S';
console.log(strObj); // 'Lee'
```

- String 래퍼 객체는 배열과 유사하게 인덱스를 사용하여 각 문자에 접근할 수 있다.
- 단, 문자열은 원시 값이므로 변경할 수 없다. (그리고 이때 에러가 발생하지 않는다.)

```
let strObj= new String(123);
console.log(strObj); // String(0:"1", 1:"2", 2:"3", length:4, ...}
```

- String 생성자 함수의 인수로 문자열이 아닌 값을 전달하면 인수를 문자열로 강제 변환

```
String(1); // -> "1"
String(true); // -> "true"
```

- new 연산자를 쓰지않고 String 생성자 함수를 호출하면 `String 인스턴스`가 아닌 `문자열`을 반환한다.

32.2 length 프로퍼티

- length 프로퍼티는 문자열의 문자 개수를 반환한다.

```
'Hello'.length; //-> 5
```

32.3 String 메서드

- **배열**에는 원본 배열을 `직접 변경하는 메서드`와 `새로운 배열을 생성하여 반환하는 메서드`가 있다.
- 하지만, **String** 객체에는 원본 String 래퍼객체를 `직접 변경하는 메서드`는 **존재하지 않는다**.
- 문자열은 변경 불가능(immutable)한 원시 값이기 때문에 String 래퍼 객체도 `읽기 전용 객체`로 제공된다.

```
const strObj = new String('Lee');

console.log(Object.getOwnPropertyDescriptors(strObj));
/*
{
    0: {value: 'L', writable: false, enumerable: true, conf
igurable: false}
    1: {value: 'e', writable: false, enumerable: true, conf
igurable: false}
    2: {value: 'e', writable: false, enumerable: true, conf
igurable: false}
    length: {value: 3, writable: false, enumerable: false,
configurable: false}
}
*/
```

다음은 사용 빈도가 높은 String 메서드이다.

32.3.1 String.prototype.indexOf

- `대상 문자열`에서 `인수로 전달받은 문자열`을 검색하여 `첫 번째 인덱스`를 반환한다.
- 검색에 실패하면 `-1`을 반환한다.

```
const str="Hello World";

str.indexOf('l'); // -> 2
str.indexOf('or'); // -> 7
str.indexOf('x'); // -> -1
```

- 2번째 인수로 검색을 시작할 인덱스를 전달할 수 있다.

```
str.indexOf('l', 3); //-> 3
```

- 특정 문자열이 존재하는지 확인할 때, 다음과 같이 사용할 수 있다.
- 하지만 ES6에서 도입된 includes 메서드를 사용하면 가독성이 더 좋다.

```
//indexOf 사용했을때
if(str.indexOf('Hello') !== -1){
}

//includes 사용했을때
if(str.includes('Hello')){
}
```

32.3.2 String.prototype.search

- 인수로 전달받은 정규 표현식과 매치하는 문자열을 검색하여 `문자열의 인덱스`를 반환한다.
- 검색에 실패하면 -1을 반환한다.

```
const str= 'Hello world';
```

```
str.search(/o/); // -> 4
str.search(/x/); // -> -1
```

32.3.3 String.prototype.includes

- 대상 문자열에 인수로 전달받은 문자열이 포함되어 있는지 확인하여 true/false로 반환한다.
- 2번째 인수로 검색을 시작할 인덱스를 전달할 수 있다.

```
const str= 'Hello world';

str.includes('Hello'); //-> true
str.includes('x'); //-> false

str.includes('l', 3); // -> true
```

32.3.4 String.prototype.startsWith

- 대상 문자열이 인수로 전달받은 문자열로 시작하는지 확인하여 true/false 반환
- 2번째 인수로 검색을 시작할 인덱스를 전달할 수 있다.

```
const str= 'Hello world';

str.startsWith('He') //-> true
str.startsWith('x') // -> false

str.startsWith(' ', 5); // -> true
```

32.3.5 String.prototype.endsWith

- 대상 문자열이 인수로 전달받은 문자열로 끝나는지 확인하여 true/false 반환
- 2번째 인수로 검색할 문자열의 길이를 전달할 수 있다.

```
const str= 'Hello world';

str.endsWith('ld'); // -> true
```

```
// str의 5자리까지(Hello)가 'lo'로 끝나는지 확인
str.endWith('lo', 5); //-> true
```

32.3.6 String.prototype.charAt

- 대상 문자열에서 인수로 전달받은 인덱스에 위치한 문자를 검색하여 반환
- 인덱스는 문자열의 범위사이의 정수여야 한다.
- 인덱스가 문자열의 범위를 벗어난 정수인 경우 빈 문자열을 반환한다.

```
const str = 'Hello';

str.charAt(0); // -> H
str.ChartAt(5); // -> ''
```

32.3.7 String.prototype.substring

- 대상 문자열에서 `첫 번째 인수`로 전달받은 인덱스에 위치하는 문자부터 `두 번째 인수`로 전달받은 인덱스에 위치하는 문자의 바로 이전 문자까지의 부분 문자열을 반환한다.
- 두 번째 인수를 생략할 경우 마지막 문자까지 반환

```
const str = 'Hello World';

str.substring(1,4); //-> ell
str.substring(1); //-> ello world
```

첫 번째 인수는 두 번째 인수보다 작은 정수여야 정상이지만,

다음과 같이 인수를 전달하여도 정상 동작한다.

1. 첫 번째 인수 > 두 번째 인수 → 두 인수는 교환된다.
2. 인수 < 0 또는 NaN → 0으로 취급
3. 인수 > 문자열의 길이 → 문자열의길이(str.length)로 취급

32.3.8 String.prototype.slice

- substring 메서드와 동일하게 동작하며, 음수인 인수를 전달할 수 있다.

- 음수인 인수를 전달하면 대상 문자열의 가장 뒤에서부터 시작하여 문자열을 잘라내어 반환한다.

```
const str = 'Hello World';

str.slice(0,5); //-> 'Hello'
str.slice(-5); // -> 'World'
```

32.3.9 String.prototype.toUpperCase

- 대상 문자열을 모두 대문자로 변경한 문자열을 반환

```
const str = 'Hello World';

str.toUpperCase(); // -> 'HELLO WORLD'
```

32.3.10 String.prototype.toLowerCase

- 대상 문자열을 모두 소문자로 변경한 문자열을 반환

```
const str = 'Hello World';

str.toLowerCase(); // -> 'hello world'
```

32.3.11 String.prototype.trim

- 대상 문자열 앞뒤에 공백 문자가 있을 경우 이를 제거한 문자열을 반환

```
const str= '   foo   ';

str.trim(); // -> 'foo'
```

32.3.12 String.prototype.repeat

- 대상 문자열을 인수로 전달받은 정수만큼 반복해 연결한 새로운 문자열을 반환한다.
- 0이면 빈 문자열을 반환(인수를 생략하면 기본값 0이 설정된다)

- 음수이면 RangeError를 발생

```
const str = 'abc';

str.repeat(); // ->''
str.repeat(2); //-> 'abcabc'
```

32.3.13 String.prototype.replace

- 대상 문자열에서 `첫 번째 인수`로 전달받은 문자열 또는 정규표현식을 검색하여 `두 번째 인수`로 전달한 문자열로 치환한 문자열을 반환
- 검색된 문자열이 여러개일 경우 첫 번째 검색된 문자열만 치환한다.

```
const str= 'Hello world';

str.replace('world', 'Lee'); // -> 'Hello Lee'
```

32.3.14 String.prototype.split

- 대상 문자열에서 첫 번째 인수로 전달한 문자열 또는 정규 표현식을 검색하여 문자열을 구분한 후 `분리된 각 문자열`로 이루어진 `배열`을 반환한다.
- 인수로 빈 문자열을 전달 → 각 문자를 모두 분리
- 인수를 생략 → 대상 문자열 전체를 단일 요소로 하는 배열 반환

```
const str= "How are you doing?";

str.split(' '); // ["How", "are", "you", "doing?"]
```

- 두 번째 인수로 배열의 길이를 지정할 수 있다.

```
str.split(' ', 3); // ["How", "are", "you"]
```